CS590R - Algorithms for communication networks
Lecture 9
Randomized Routing

Lecturer: Gopal Panduranagan
Scribe: Ossama M. Younis
Department of Computer Sciences
Purdue University

1 Oblivious routing

Definition: Routing in which the path of one packet does not depend on the source and destination of any other packet in the system. Oblivious routing satisfies the following property: The route followed by a packet q at source node i depends on its destination $dst(i)$, and not on any $dst(j)$, for any $j \neq i$. Thus, an oblivious routing algorithm specifies, for each pair $(i, dst(i))$, a route between node i and node $dst(i)$ [3].

Next, we try to compute the lower bound for deterministic oblivious routing. The following theorem computes the lower bound on the number of routes that may traverse a vertex for some permutation.

Theorem 1 *For any deterministic oblivious permutation routing algorithm on a network of N nodes, each of out-degree d, there is an instance of permutation routing where some vertex v is traversed by $\Omega(\sqrt{N/d})$ routes (packets) [1].*

The proof of this theorem can be found in [1]. It is beyond the scope of these notes. One result of the above theorem is the following. For a hypercube, every node has $n = log\ N$ directed edges leaving it. Each edge incident on a node is associated with a distinct bit position in the node label. The theorem says that for any deterministic oblivious routing algorithm on the hypercube, there is a permutation requiring $\Omega(sqrtN/n)$ steps.

A more precise computation of the lower bound on deterministic oblivious routing is provided in the following theorem.

Theorem 2 *For any N-node network with maximum degree d, and any deterministic oblivious packet routing algorithm, there is a permutation that requires $\Omega(\sqrt{N/d^3})$ steps.*

Proof. Given a network and a deterministic oblivious algorithm we construct a permutation π on the N addresses, such that in routing π there is some vertex v that is traversed by at least $\sqrt{N/d}$ packets. Since the degree is bounded by d, it implies that routing takes $\Omega(\sqrt{N/d^3})$.

A deterministic oblivious algorithm is defined by N^2 routes, where $P_{u,w}$ gives the route taken by a packet from u to w.

To construct a "bad" permutation we need a vertex x such that for many destinations $v_1, \ldots, v_k$ many origins $w_1, \ldots, w_k$ send their packets through x.

Let S_k^v be the set of vertices such that at least k paths to v enter each of these vertices.

Lemma 1

$$|S_k^v| \geq \frac{N}{d(k-1)+1}$$

Proof.

- Clearly $v \in S_k^v$ since $N-1$ paths to v enter v.
- All paths from an origin outside S_k^v must enter S_k^v.
- The set S_k^v has no more than $d|S_k^v|$ direct neighbors outside the set.
- By the definition of S_k^v no more than $k-1$ paths to v can enter any vertex that is not in the set.
- Thus, at most $|S_k^v|d(k-1)$ paths can enter the set S_k^v. The remaining paths must start inside the set.
- Thus,

$$|S_k^v| + |S_k^v|d(k-1) \geq N$$

Therefore, it follows that:

- We need a vertex x that is included in many S_k^v for many v's.

$$\sum_{v \in V} |S_k^v| \geq \frac{N^2}{d(k-1)+1}$$

- Since there are only N vertices, each vertex is "on average" in $\frac{N}{d(k-1)+1}$ sets.
- Thus, there is at least one vertex x, and $l = \frac{N}{d(k-1)+1}$ vertices $v_1, \ldots, v_l$ such that $x \in S_k^{v_i}$ for $i = 1, \ldots, l$.
- For each of the l vertices there are k origins that uses x on the path to v_i.
- Fix $k = \sqrt{N/d}$. then $l \geq \sqrt{N/d}$.
- We can construct a permutation with k paths that traverse vertex x.

2 The Butterfly Network

Although a hypercube is computationally powerful, the degree of its nodes is unbounded and it grows with its size. One topology that has been devised to circumvent the difficulty of node degrees in hypercubes is the butterfly network. In this section, we give some brief description of properties of a butterfly network [2].

2.1 Definition

An r-dimensional butterfly has $(r+1)2^r$ nodes and $r.2^{r+1}$ edges. The nodes correspond to pairs $< w, i >$, where i is the dimension or level of the node ($0 \leq i \leq r$) and w is an r-bit binary number that denotes the row of the node. Two nodes $< w, i >$ and $< w', i' >$ are linked by an edge if and only if $i' = i + 1$ and either:
(1) w and w' are identical, or
(2) w and w' differ in precisely the i^{th} bit.

If w and w' are identical, then the edge is said to be a *straight edge*. Otherwise, the edge is a *cross edge*. Refer to figure 1 for a three-dimensional butterfly. In this figure, level 3 nodes are removed, which results in two disjoint two-dimensional butterflies, one consisting of even rows (solid lines), and one consisting of odd rows (dashed lines).

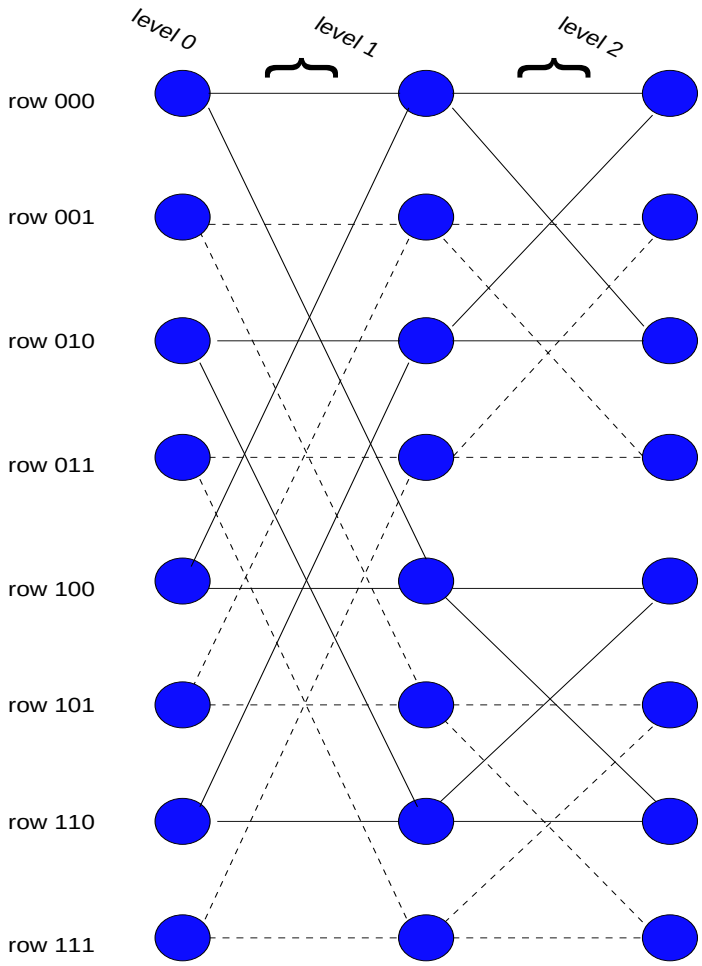

Figure 1. A three-dimensional butterfly with level 3 nodes removed

2.2 Properties

The butterfly and hypercube networks are similar in structure. The i^{th} node of the r-dimensional hypercube corresponds to the i-th row of the r-dimension butterfly. In effect, the hypercube is a folded up butterfly [2]. The butterfly network has many nice properties: (1) it has a simple recursive structure, (2) The level 0 node in any row w is linked to the level r node in any row w' by a unique path of length r, and (3) the butterfly network has a large bisection width, like hypercubes. Its bisection width is $\theta(N/logN)$.

2.3 Wrapped Butterfly

For computational purposes, the first and last levels of the butterfly network are sometimes merged into a single level. In particular, node $< w, 0 >$ is merged into node $< w, r >$ for each w. The result is an r-level graph with $r.2^r$ nodes, each with degree 4. Figure 2 shows a 3-dimensional wrapped butterfly network. One of the important properties of wrapped butterfly networks is provided in the following theorem [2].

Theorem 3 *Given an N-node wrapped butterfly with at most one packet at each node, and any permutation $\pi : [1, N] \rightarrow [1, N]$, there is a way of routing the packets through the network so that within $3logN$ steps, the packet that started at node i (if any) is located at node $\pi(i)$ for $1 \leq i \leq N$. In addition, the movement of packets*

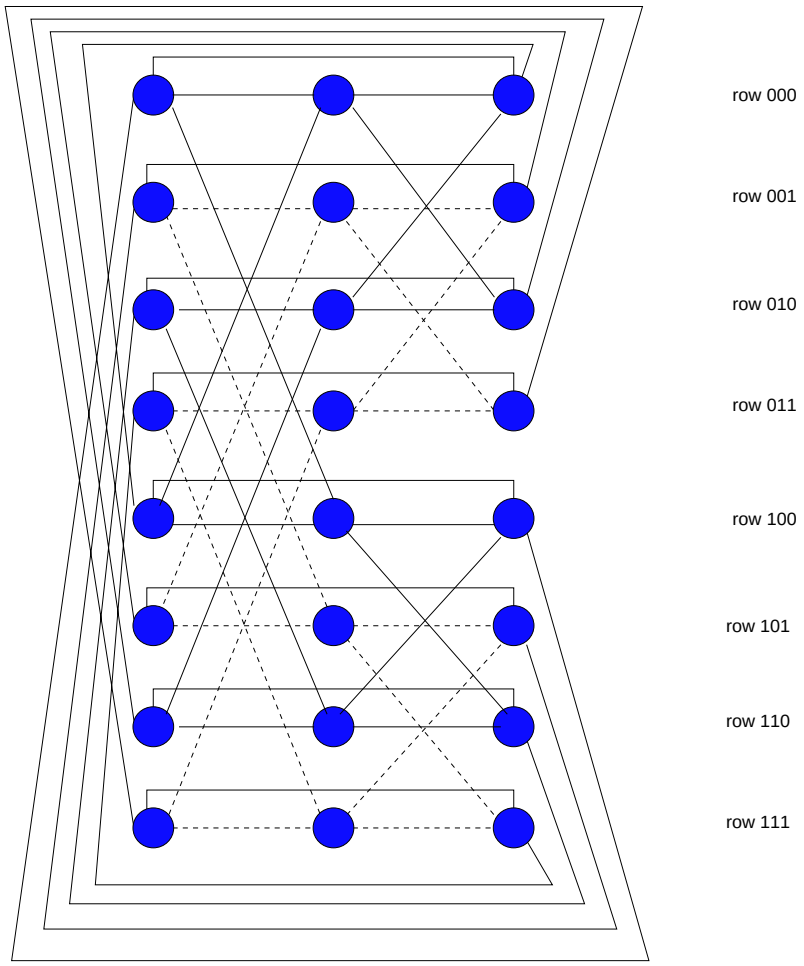

Figure 2. A three-dimensional wrapped butterfly

is such that at most one packet traverses any wire in any step, and at most three packets reside in any node at any time.

Proof of the theorem can be found in [2].

3 The mesh topology

The last architecture that we discuss in this lecture is the mesh-topology. A $k-d$ mesh is a d-dimensional array, with k nodes along each dimension. The hypercube is a special case of a $k-d$ mesh, with 2 nodes along each dimension and $logN$ dimensions.

Routing on a 2-D mesh can be carried out using the following simple greedy algorithm:
(i) Route along one dimension until destination is reached, then
(2) start on the other dimension.
This approach yields routes in 2*$\sqrt{N}$ steps, which is optimal. However, some buffers may pile up O($\sqrt{n}$) packets.

To solve this problem, a randomized approach (3-phase routing) can be used as follows.
(1) Divide the mesh into $\sqrt{N}/logN$ on one dimension.
(2) A node first picks a random node along its $\sqrt{N}/logN$ part, and send the packet to this random choice.
(3) At the randomly chosen node, the greedy routing approach is followed as specified above.

Table 1. Switching approaches

	Store and forward	*Circuit switching*
Approach	Messages are conveyed as *packets*. An entire packet can reside at a routing switch. Packets are sent from a queue of one switch to queue of another switch until they reach their destinations. At most one packet is transmitted on a link in each step. A link can be shared by different sessions using that link, but the sharing is on a demand basis, rather than a fixed allocation basis.	When a session (or connection) is initiated, bandwidth is allocated along a path connecting the two endpoints for the whole duration of the connection.
Advantages	(1) Faster adaptation to find "better" path for each packet. (2) More fault-tolerant. (3) More responsive to interactive real-time applications.	(1) Connection setup is done only once. (2) Smaller routing table. (3) Simpler switch design. (4) In-order delivery of packets.
Drawbacks	(1) Larger routing tables. (2) More complex switch design. (3) Increased processing power requirement. (4) Possible out-of-order packet delivery.	(1) For long connections, resources may be wasted. (2) Goodness (quality) of routes may change. (3) High initial setup cost. (4) Less fault-tolerant.

4 Switching approaches

Routing strategies have followed one of two major switching approaches *store and forward switching*, which was the precursor for packet switching, and circuit switching, which is the technology currently deployed in the world's telephone network. Table 1 defines and compares the two approaches. Note that, the original store and forward approach used to send the whole message. Packet switching uses similar approach but the message is divided into smaller units. In the next discussion, we will use the two terms to mean packet switching. Refer to [4] for more details.

4.1 More definitions

Virtual circuit routing is store-and-forward switching in which a particular path is set up when a session is initiated and maintained during the life of the session. Like circuit switching, but links are shared by sessions on a demand basis.

Dynamic routing or *datagram routing* is store and forward switching in which each packet finds it own path through the network using the current information available at the nodes visited.

5 Side Notes

The following inequality is a special instance of the inclusion-exclusion principle in set-theory. **Boole's inequality (or Union bound)** is defined as follows. If $A_1, A_2, \ldots, A_n$ are arbitrary events, then

$$Pr(\bigcup_{i=1}^{n} A_i) \ \leq \ \sum_{i=1}^{n} Pr(A_i)$$

References

[1] A. Borodin and J. Hopcroft. Routing, Merging, and Sorting on Parallel Models of Computation. In *Journal of Computer and System Sciences*, pages 130–145, 1985.

[2] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays.Trees.Hypercubes*. Morgan Kaufmann Publishers, 1992.

[3] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[4] L. L. Peterson and B. S. Davie. *Computer Networks*. Morgan Kaufmann Publishers, Second edition, 2000.